34.3

Pre-layout Wire Length and Congestion Estimation

Qinghua Liu
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-5678
qinghual@ece.ucsb.edu

Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-2721
mms@ece.ucsb.edu

Abstract

In this paper, we study the pre-layout wire length and congestion estimation. We find that two structural metrics, *mutual contraction* and *net range*, can be used to predict wire lengths. These metrics have different application ranges and complement each other. We also propose a new metric, the *structural pin density*, to capture the peak routing congestion of designs. Larger *maximum pin densities* usually lead to larger peak congestions in circuits with similar average congestions. We demonstrate experimentally very good correlation of our pre-layout measures with post layout interconnect lengths. We also isolate the structural netlist properties which cause the peak congestion.

Categories and Subject Descriptors

J.6 Computer-Aided Engineering-Computer-aided Design (CAD)

General Terms

Algorithms

Keywords

Wire Length, Congestion, Prediction

1. INTRODUCTION

Interconnects play a crucial role in the overall performance of modern VLSI systems. Early wire optimization is needed to achieve the required performance, routability, and power. The cost functions employed during traditional logic synthesis do not take interconnects into consideration, which may lead to sub-optimal designs. On the other hand, physical synthesis with fixed circuit structure has limited improvement space. For this reason, physical and logic co-synthesis have attracted a lot of research effort in recent years. Integrating logic synthesis and physical design steps will be possible only if we understand how decisions made at the logic synthesis level affect placement and routing results.

DAC 2004, June 7-11, 2004, San Diego, California, USA

Previous works on physical and logic co-synthesis can be grouped into three categories: layout-driven logic synthesis, local netlist transformations, and metric-driven structural logic synthesis. Layout-driven logic synthesis [12][17][18] starts with creating an initial placement of the technology-independent netlist and uses placement coordinates of the objects to improve the synthesis flow. Such methods work with very inaccurate placement model since final placement is likely to change substantially after synthesis. Local-netlist-transformation methods [14][16] directly operate on a placed netlist. Targeting different objectives such as timing, or power consumption, critical parts of the circuit are extracted with accurate layout information. Re-synthesis and re-placement are used to transform them. These approaches work on correct net-lengths estimates and apply local transformations to preserve the existing placement as much as possible. This limits the optimization potential. More recently, the metrics such as *adhesion* [11] and *distance* [10] have been proposed to capture routability of designs. Those metrics were applied in logic synthesis to obtain more routable designs. We refer to this approach as *metric-driven structural logic synthesis*.

We believe that the first step towards guiding logic synthesis in producing layout-friendly results, is the ability to predict layout characteristics from the netlist structure. In this paper we focus on pre-layout wire length and congestion estimations. We find that *mutual contraction* and *net range*, two structural metrics, can be used to predict wire lengths in the final layout. They have different application ranges and complement each other well. We propose also a new metric, the *structural pin density*, to capture the peak routing congestion. This metric is much simpler to calculate than adhesion.

The paper is organized as follows. In Section 2 we discuss wire length prediction using multiple metrics. We analyze the quality of wire length predictions by *mutual contraction*, *connectivity*, *edge separability* and *net range*. In Section 3 we study the relationship between netlist structure and routing congestion. We introduce the *structural pin density*, which serves as a second-order congestion-estimation metric. Section 4 concludes the paper.

2. PRE-LAYOUT WIRE LENGTH PREDICTION

2.1 Previous work

Statistical wire length prediction has been studied for a number of years. Most of the papers [6][20] in this category follow the pioneering work of Donath in 1979, and make wire length predictions based on Rent's rule. In [19], Pedram and Preas estimate interconnect lengths by considering all possible distributions of pins in rows.

Very few papers address the individual wire-length prediction. In [13], the authors show that individual wire lengths depend on placement algorithms. Wire-length predictions accurate for one placement flow may be inaccurate for another. To solve this problem, the authors embed the wire length prediction into their FPI [8] placement flow and predict individual wire-lengths during the clustering step. This approach can make predictions only on intra-cluster nets, which tend to be short in the final placement. Another solution is proposed by Bodapati and Najm [1]. Their method requires building black-box models for the place-and-route tools, which is a time-consuming task.

In this paper we propose a new type of wire length prediction. We consider nets which have similar characteristics in terms of some structural metric. Based on the numerical values of those metrics, we can predict that nets in one group have a tendency to be longer or shorter than the members of another group. We call this kind of prediction, a *semi-individual prediction*. In this paper we restrict our discussion only to such predictions.

2.2 Predicting wire lengths of 2-pin nets

Net lengths usually increase with an increase of the pin number. The correlation is good, but this measure cannot capture wire length variations among nets with the same number of pins. Typically, in real circuits, more than 50% of nets are 2-pin nets, so we begin our discussion from those nets.

2.2.1 Predicting lengths of 2-pin nets

Several metrics, such as *connectivity* [7], *edge separability* [5], and *mutual contraction* [9], have been proposed to guide the clustering process in multi-level partitioning, or placement frameworks. In this paper, we apply them to predict wire lengths of 2-pin nets, and compare their efficiencies.

We model multi-pin nets by cliques. A connection $c(u,v)$ represents all edges connecting nodes u and v. The weight of a connection, $w(u,v)$, is a sum of edge-weights of the edges in $c(u, v)$. A node v is node u's neighbor if the connection $c(u,v)$ exists. $A(v)$ is the area-cost of a node v. Traditionally, a weight in a clique representing a multi-pin net is assumed to be:

$$w(e) = \frac{1}{d(i)-1} \qquad \text{(EQ 1)}$$

where $d(i)$ denotes a degree of a net i. *Connectivity* is a measure of contraction between two connected nodes u and v. EQ2 expresses connectivity in our terminology:

$$c_c(u, v) = \frac{w(u, v)}{A(u) \cdot A(v) \cdot (d(u) - w(u, v)) \cdot (d(v) - w(u, v))} \qquad \text{(EQ 2)}$$

Edge separability is defined for each connection $c(u, v)$ in a graph G. $\lambda(c)$ is the minimum cut-size among all the cuts separating nodes u and v in G. From this definition it follows that:

$$w(c) \le \lambda(c) \le min\{d(u), d(v)\} \qquad \text{(EQ 3)}$$

Intuitively, the larger $\lambda(c)$ is, the more contracted u and v are, since separating u and v causes a greater cut-size penalty. Computing $\lambda(c)$ for each connection c in a graph is very time-consuming, [5] proposes an algorithm to compute a tight lower bound $q(c)$ for $\lambda(c)$ such that:

$$w(c) \le q(c) \le \lambda(c) \qquad \text{(EQ 4)}$$

Based on the metric q(c), a connection-ranking formula given by EQ5 is suggested in [5] and used for greedy clustering:

$$r(c) = \frac{q(c)}{min\{d(u), d(v)\}} \qquad \text{(EQ 5)}$$

Mutual contraction of a connection(u,v) has been proposed in [9]. For multi-pin nets, the authors propose a modified clique-edge weight:

$$w'(e) = \frac{2}{(d(i)-1)\,d(i)} \qquad \text{(EQ 6)}$$

and introduce the relative weight of a connection:

$$w_r(u, v) = \frac{w'(u, v)}{\sum w'(u, x)} \qquad \text{(EQ 7)}$$

In EQ7, $w'(u, x)$ is a weight of the connection (u, x), the summation is taken over all nodes x adjacent to u. The product of $w_r(u,v)$ and $w_r(v,u)$ is a measure of the mutual contraction of a connection (u,v):

$$c_p(x, y) = w_r(x, y)\, w_r(y, x) \qquad \text{(EQ 8)}$$

Connections with strong contractions are most likely to end up having short lengths.

2.2.2 Comparing the metrics

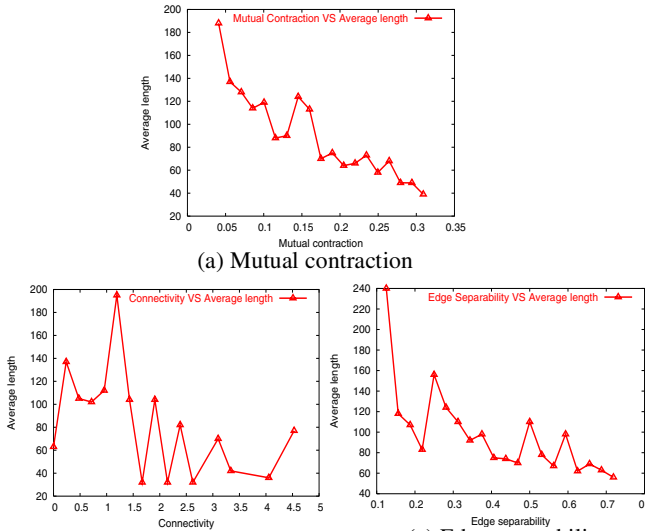
We experiment with IBM [23] and LGSyn93 benchmark suites. Our LGSyn93 suite consists of 21 benchmarks selected from [26]. We synthesize the netlists of LGSyn93 benchmarks by *script.delay* and *map -n 1 -AFG* commands in SIS using a standard cell library translated from a commercial 0.13um library. The mapped circuits in LGSyn93 suite range from 617 to 12562 cells. In the rest of the paper, all placements of the circuits are generated by Capo8.7, unless indicated otherwise.

We explain our experiments for the mutual contraction case. Experiments with other metrics are run in the same fashion. We calculate the mutual contractions of all 2-pin nets, and for each benchmark we determine the range $[C_{min}, C_{thr}]$. C_{thr} is a threshold value of mutual contraction determined such that the summation of net lengths (after placement), whose mutual contractions are larger than C_{thr}, is smaller than 5% of the total wire length. We partition $[C_{min}, C_{thr}]$ into 20 equal-sized bins:

$$[A_i, B_i],\ i=1,2...20,\ A_1=C_{min}\ B_{20}=C_{thr}$$

We assign nets to bins based on their mutual contractions and then calculate the average length of nets in each bin.

The results for *ibm01* are shown in Figure 1. The x-axis corresponds to bins sorted by their left boundary values. The y-axis shows the average net lengths for the placed circuit. Figure 1(a) shows the mutual contraction case, 1(b) the connectivity, and 1(c) the edge separability. From this figure we observe that all these metrics are correlated with wire lengths.



(a) Mutual contraction

(b) Connectivity (c) Edge separability

Figure 1: Placed wire length vs. prediction metrics on *ibm01*

Table 1. Average *c-ratio* and *r-square* values for IBM- and LGSyn93-benchmarks on different metrics

Benchmarks	Mutual Contraction		Connectivity		Edge Separability	
	c-ratio	r-square	c-ratio	r-square	c-ratio	r-square
IBM	0.903	0.701	0.710	0.260	0.811	0.528
LGSyn93	0.858	0.635	0.776	0.327	0.778	0.430

We will refer to a pair (metric, length) as a *node*. We define a *correlation ratio* (*c-ratio*) to quantify the correlation between a metric and wire length by the following equation:

$$c-ratio = 1 - \frac{ViolationNodesPairs}{TotalNodePairs} \quad \text{(EQ 9)}$$

In the above equation, *TotalNodePairs* denotes the total count of node pairs. If there are *n* (metric, length) nodes, there are $\binom{n}{2}$ *TotalNodePairs*. *ViolationNodePairs* denotes the number of node pairs violating the correlation. Take two nodes in Figure 1(a) for example. If one node has larger mutual contraction and larger wire length than the other, then this pair violates the negative correlation between mutual contraction and wire length. *C-ratio* is a number between 0 and 1. A larger *c-ratio* corresponds to a better correlation or better predictability of the metric. We also compute *r-square,* the linear regression fit of the wire length for each metric. In Table 1, we show the average c-*ratio* and *r-squar*e for IBM and LGSyn93 benchmarks for different metrics. We note that the correlation between the mutual contraction and wire length is the best.

We have also verified that the good predictability of the mutual contraction holds for different placers. We repeated the experiments on IBM benchmarks using Capo8.7 [2], Dragon [23] and mPL2.1 [3].

2.3 Limitations of mutual contraction

The IBM- and LGSyn93-benchmarks have Rent's exponent around 0.6. So they all have similar interconnection complexity.

PEKO [4] benchmarks are a special class of low Rent's exponent circuits. In the optimal solutions of PEKO benchmarks, all nets are local and their lengths are the minimum possible. Nets with the same pin numbers have exactly the same lengths, though their mutual contractions are usually different. Mutual contraction is not correlated well with placed wire-lengths on PEKO benchmarks.

Using *gnl* [21], we built a suite of synthetic circuits with different Rent's exponents. Each of these circuits has 40,000 cells and the same number of nets. We place those circuits and collect wire lengths and mutual contraction values for 2-pin nets. We list the *c-ratios* and *r-squares* for all synthetic benchmarks in Table 2.

Table 2. Predictive power of mutual contraction on circuits with different Rent's exponents

r	*c-ratio*	*r-square*
0.3	0.531	0.083
0.4	0.667	0.131
0.5	0.760	0.377
0.6	0.879	0.648
0.7	0.953	0.741
0.8	0.857	0.673

In Table 2, the first column lists the Rent's exponents. The results suggest that mutual contraction does not work well on low Rent's exponent circuits.

In [9], mutual contraction has been extended from 2-pin to multi-pin nets. Here we examine how mutual contraction works on multi-pin nets.

In Table 3 we compare the predictability of mutual contraction on 2-pin and multi-pin nets by *c-ratio* and *r-square*. In the table, the data are computed as an average for 18 IBM benchmarks.

Table 3. Predictability using mutual contraction on nets with different pin numbers

pin number	c-ratio	r-square
2	0.903	0.701
3	0.529	0.005
4	0.631	0.118

From the table, we observe that the correlation between mutual contraction and wire lengths of multi-pin nets is not as good as in the case of 2-pin nets. We need other metrics to predict accurately wire lengths of multi-pin nets. In the next section, we will show that *net_range* is a good candidate.

2.4 Net range

The *fanout-range-of-a-node* metric was first proposed in [22]. It is defined as a circuit depth spanned by the fanout nodes.

In [22] the authors show that the net lengths increase with an increase in the fanout ranges. The correlation is especially good for nets with low fanout ranges. This conclusion may be misleading. Nets with larger pin numbers tend to have larger fanout ranges. So the correlation that the authors have observed may be actually caused by the pin number instead of fanout range. Fanout range cannot distinguish two cases in Figure 2 - both nets have fanout range 1. It is intuitive that in the final layout, the right net will most likely be longer than the left net.

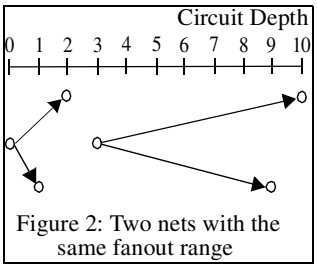


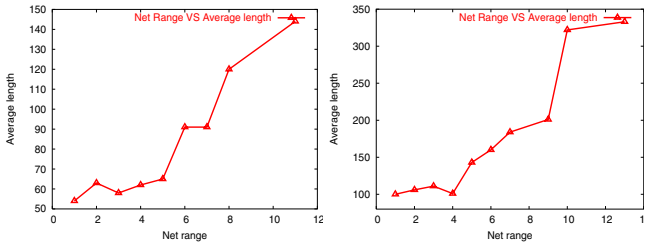
Figure 2: Two nets with the same fanout range

Based on these observations, we define *net range* as a circuit depth spanned by all terminal nodes of a net. From the definition, the left net in figure 2 has net range of 2, and the right one has a net range of 7. We use the net range metric to predict wire lengths of nets with the same pin number. It is intuitive that larger *net-range*-nets usually correspond to longer net lengths. In this section, we will examine the predictive power of the *net range* metric.

In IBM benchmarks, the fanin and fanout information are incomplete and we cannot traverse circuits to assign circuit depths to the nodes. Because of that, we cannot calculate net ranges for nets in IBM benchmarks. In this section we test the *net range* as a prediction metric. We use the LGSyn93 benchmark suite and the synthetic benchmark suite generated by *gnl*.

First, we place the LGSyn93 benchmarks and plot the relationship between the net range and wire length. We take

the benchmark *seq* as an example and show the results for 3-pin and 4-pin nets.

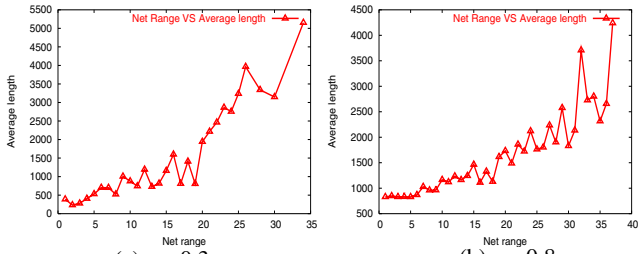


(a) 3-pin nets (b) 4-pin nets

Figure 3: Placed wire lengths vs. net range for benchmark *seq*

In Figure 3, on the *x*-axis we have *net range* and on the *y*-axis we have the average lengths of nets corresponding to those *net ranges*. We do not show the results for 2-pin nets because most of the 2-pin nets have a net range of 1. In other words, net range is too crude a measure for 2-pin nets. We observe that *net range* works well on nets with pin numbers larger than 2. Larger *net ranges* correspond to longer net lengths.

In section 2.3, we observed that the predictability of mutual contraction varies with interconnection complexity. Here we use the same synthetic benchmark suite and test the predictability of a *net range*. In Figure 4, we show the results for 3-pin nets for circuits with Rent's exponent 0.3 (very low) and 0.8 (large). We observe that *net range* works well on the whole spectrum of Rent's exponent circuits.



(a) r = 0.3 (b) r = 0.8

Figure 4: Placed wire length vs. net range on circuits with different Rent's exponents

2.5 Summary

It appears that accurate pre-layout wire length prediction is not possible with just one of the examined metrics. Among different metrics, *mutual contraction* and *net range* show the best prediction efficiency. These two metrics have different application ranges. *Mutual contraction* works well only on 2-pin nets in high Rent's exponent circuits. *Net range* works well on both high Rent's exponent and low Rent's exponent circuits, but since it is too coarse for two-pin nets, we apply it only to multi-pin nets. These two metrics are complementary, and jointly work as accurate pre-layout wire length predictors.

3. CAPTURING ROUTING CONGESTION FROM NETLIST STRUCTURE

3.1 Previous work

The increasingly large sizes of modern circuits lead to increasingly long design turn-around time. This problem is often caused by the routing congestion which has to be resolved before reaching convergence. Predicting and removing routing congestion has been attracting a great deal of attention. Research efforts focus on determining routable placements.

Accurate information about routing congestion is not available until after detailed routing, which is too late to make significant changes. Therefore, it is desirable to have congestion estimations earlier in the design cycle. In [15], the authors propose a stochastic congestion estimation algorithm for a placed netlist. In [24], Rent's rule is used to quantitatively estimate the maximum congestion prior to the placement stage. None of these methods can be applied to optimize for congestion since it is difficult to capture Rent's exponent during structural transformations. Recently, a new measure called *adhesion* [11] has been proposed to predict routability of logic networks. Adhesion has been applied to logic synthesis to improve routability. It is not clear how to determine and update it in a computationally efficient manner.

In this section, we propose a new metric called *structural pin density* to predict the peak congestion for designs with similar average congestion. This metric is very simple and can be measured very efficiently.

3.2 Structural pin density

We cover the chip area with a matrix of equal-sized bins. We define routing congestion as the number of nets which after global routing cross a bin boundary. Congestion and routing requirements are very closely related. When considering congestion, we need to look at both average congestion and peak congestion. We define *routing area ratio* as the ratio of the total wire length and chip area:

$$R_{rar} = \frac{TWL}{Area} \quad \text{(EQ 10)}$$

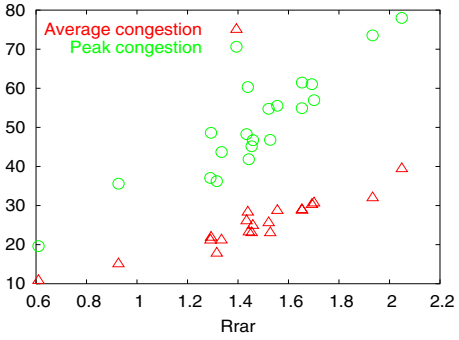


Figure 5: R_{rar} VS Congestion

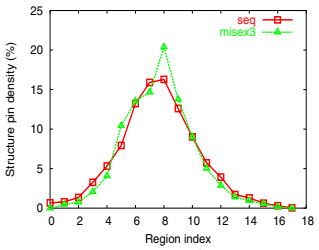
We use the same LGSyn93 benchmark suite as in section 2, and in Figure 5 we plot the relationship between R_{rar} and congestion in placed and routed designs. In these experiments the chips have aspect ratio 1. We obtain the congestion maps from placements using the congestion plotter [25].

In Figure 5, the *x*-axis measures R_{rar} and *y*-axis measures congestion. The circles represent peak congestions and triangles represent the average congestion. We observe that there is a correlation between the R_{rar} and congestion.

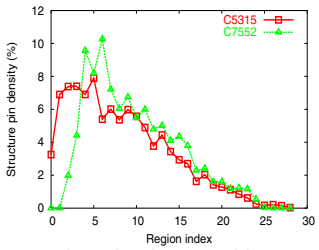
Looking at Figure 5, we also note that a larger average congestion usually corresponds to a larger peak congestion. However, in some cases, circuits with similar average congestion may have very different peak congestions. To capture the peak congestion from the netlist structure more precisely, we need a second order metric.

We define *node level* as the circuit depth of the node. Also we define *net level* of a net *i* by the formula:

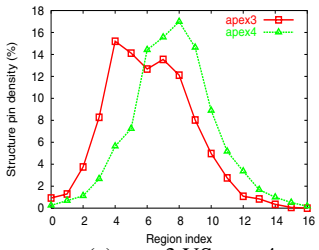
$$NetLevel(i) = \frac{\sum_{Node\ j \in Net\ i} NodeLevel(j)}{NetDegree(i)} \quad \text{(EQ 11)}$$



(a) seq VS misex3



(b) C5315 VS C7552



(c) apex3 VS apex4

Figure 6: Comparing the *structural pin density* distribution for different benchmarks

Using circuit depth, we partition a circuit from PIs to POs into n regions (slices) of the same depth. For example, we may partition a circuit of depth 20 into 5 regions, each of depth 4. A partition is *maximal* if all regions have depth 1. We associate each net with a region covering this net's level. After that, we define the *structural pin density* of a region *i* by:

$$PinDensity(i) = \frac{\sum_{Net\, j \in Region\, i} NetDegree(j)}{TotalPinNumber} \quad \text{(EQ 12)}$$

The *structural pin density* is a ratio between the total number of pins on the nets in a region *i,* and the total pin count in the whole circuit. The *maximum pin density* is the largest *structural pin density* among all the regions in a maximal partition. Intuitively, larger maximum pin density suggests that interconnects are more entangled in some region of the circuit. We conjecture that a larger *maximum pin density* will lead to a larger peak congestion in circuits with similar average routing congestion or R_{rar}.

Circuits with larger circuit depths tend to have smaller maximum pin density. So when comparing the maximum pin density between two circuits with different circuit depths, we normalize the maximum pin density values to the average pin density.

3.3 Maximum pin density vs. peak congestion

To verify our conjecture, we first observe three benchmark pairs selected among the 21 benchmarks used in obtaining data in Figure 5. Pertinent measures for those benchmark-pairs are listed in Table 4.

Table 4. Benchmark paris with similar average congestions but very different peak congestions

Bench pairs	1		2		3	
	seq	misex3	C5315	C7552	apex3	apex4
R_{rar}	1.6529	1.6536	1.4427	1.2933	1.7012	1.6928
ave_c	28.874	28.753	23.189	21.826	30.602	30.212
peak_c	54.914	61.446	41.837	48.608	56.922	61.063

In this table, *ave_c* denotes the average congestion and *peak_c* denotes the peak congestion. Benchmarks in each pair have similar average congestions but very different peak congestions. For example, *seq* has a little larger average congestion but significantly smaller peak congestion than *misex3*. In Figure 6 we plot the pin density distribution in the maximal partitions of those benchmarks.

From Table 4 and Figure 6, we observe that in each benchmark pair, the benchmark with a larger maximum pin density has a larger peak congestion in the final placement.

To further examine the *maximum pin density* measure, we first build different benchmark suites from the same original LGSyn93 benchmarks by applying different logic synthesis flows. In SIS, we optimize the netlists by the scripts *rugged*, *boolean* and *algebraic*. Then we perform mapping using commands *map -m 0*, *map -m 0.5* and *map -n 1 -AFG*. These 9 combinations of logic synthesis and technology mapping steps result in 9 different versions of each original benchmark circuit. Together with the circuits used in previous experiments, we have 10 benchmark suites and we refer to them as LGS01 through LGS10. Each suite contains different mutants of the same initial benchmark examples. We compute the maximum pin density and the average congestion and predict the peak congestion in the following way. For each benchmark suite, we first determine for each pair of circuits their relative average congestions. Our estimations are computed based on the *c-ratio* method defined in EQ9. In the first experiment a node is a pair (average congestion, peak congestion). We validate the relationship between the average and peak congestion by computing the *c-ratios* (second column in Table 5). In the second experiment when average congestions differ less than 5%, a node is (*maximum pin density*, peak congestion); otherwise a node is (average congestion, peak congestion). Again we compute *c-ratios* and list them in column 3 of Table 5. We repeat the same experiments by replacing average congestion by R_{rar}. The results are shown in columns 4 and 5 in Table 5.

In Table 5, *ave_c* denotes the average congestion and *mp* stands for the *maximum pin density*. We observe that in case when two benchmarks have similar average congestions or similar R_{rar} values, the difference between their peak congestions can be well explained by the *maximum pin density*. The *Maximum pin density* serves well as a second order metric for peak congestion prediction.

Table 5. Maximum pin density vs. peak congestion as measured by *c-ratio*

Bench	ave_c	ave_c+mp	R_{rar}	R_{rar}+mp
LGS01	0.879	0.947	0.805	0.879
LGS02	0.862	0.929	0.732	0.832
LGS03	0.863	0.916	0.742	0.816
LGS04	0.889	0.942	0.842	0.921
LGS05	0.847	0.926	0.742	0.821
LGS06	0.884	0.921	0.737	0.842
LGS07	0.886	0.963	0.879	0.947
LGS08	0.823	0.889	0.758	0.804
LGS09	0.889	0.924	0.772	0.848
LGS10	0.901	0.930	0.871	0.936
Ave	0.872	0.929	0.788	0.854

3.4 Relationship between logic synthesis and congestion distribution

Logic synthesis has a significant impact on a netlist structure. In this section, we examine the influence of logic synthesis on congestion. We consider only the congestion distribution.

By applying different logic synthesis flows on the same benchmark, we get different mutants of the same circuit. In our examples we optimize each circuit by the *script.rugged* followed by one of the technology mapping commands: *map -m 0* or *map -n 1 -AFG*. After placing the netlists, we compare their congestion maps. Here we use the mPL2.1 placer. For each benchmark we divide the chip area into bins such that each version of the same example is covered by the same number of rows and columns of bins. Since we are interested in high congestion areas, for each example, we determine the indices of the 10% of the highest congestion bins. The number of selected bins *n* is the same for all circuit-pairs. For any two congestion maps corresponding to mutants of the same circuit, we find those *m* bins whose indices are present in selected bins in both circuits. We define the *overlapping ratio* as *m/n*. The overlapping ratios for the congestion maps are shown in the third column of Table 6. We repeat the same experiment on circuits obtained by optimizing the initial benchmark using *script.algebraic* and *script.boolean* followed by the same mapping command *map -n 1 -AFG*. The overlapping ratios are shown in the fourth column of Table 6.

Table 6. Overlapping ratio for congestion maps generated by different logic synthesis flows

Bench	# nodes	experiment 1	experiment 2
C3540	1743	0.333	0.211
C5315	3212	0.286	0.273
C6288	4833	0.083	0.220
alu4	2367	0.381	0.333
apex4	1384	0.429	0.205
ave	2708	0.302	0.248

In Table 6, *# nodes* denotes the number of nodes in initial circuits before optimization and technology mapping. Due to page limitations, we show results only for five benchmarks. Our experiments suggest that logic synthesis has a significant impact on the congestion distribution of the final layout. Guiding logic synthesis by congestion maps obtained from un-optimized netlists would be misleading.

4. CONCLUSIONS

In this paper, we studied the pre-layout wire length prediction and congestion estimation. We found that *mutual contraction* and *net range*, the two structure-level metrics, predict wire lengths quite well. These two metrics have different application ranges and complement each other in accurate wire length predictions. We also analyzed the relationship between netlist structure and routing congestion and proposed a new second-order metric, called *structural pin density* which captures the relative peak congestions. Larger *maximum pin densities* usually lead to larger peak congestions in circuits with similar average congestions. Future work includes logic synthesis based on wire length predictions and congestion driven logic optimization.

5. ACKNOWLEDGEMENTS

This work was supported in part by the California MICRO Program through IBM and Fujitsu Laboratories of America and in part by the MARCO/Darpa Giga Scale Research Center. We acknowledge the equipment grant from Intel.

6. REFERENCES

[1] S.Bodapati and F.N.Najm, "Pre-layout estimation of individual wire lengths", ACM Intl. Workshop on System-Level Interconnect Prediction, pp.93-98, 2000.

[2] A.E.Caldwell, A.B.Kahng and I.L.Markov, "Can recursive bisection alone produce routable placements", Design Automation Conference, pp.260-263, 2000.

[3] T. F. Chan, J. Cong, J. Shinnerl and K. Sze, "An enhanced multilevel algorithm for circuit placement," Proc. of ICCAD, 2003.

[4] C.C.Chang, J.Cong and M.Xie, "Optimality and scalability study of existing placement algorithms", ASP-DAC, pp.621-627, 2003.

[5] J.Cong and S.K.Lim, "Edge separability based circuit clustering with application to circuit partitioning", ASP-DAC, pp.429-434, 2000.

[6] J.A.Davis, V.K.De, and J.D.Meindl, "A stochastic wire-length distribution for gigascale integration (GSI)-Part I: Derivation and validation", IEEE Trans. on Electron Devices, vol.45, pp.580-589, 1998.

[7] S.Hauck and G.Borriello, "An evaluation of bipartitioning techniques", IEEE Trans. on CAD, vol 16, No.8, 1997.

[8] B.Hu and M.Marek-Sadowska, "Fine-granularity clustering for large-scale placement problems", Proc. of ISPD, pp.67-74, 2003.

[9] B.Hu and M.Marek-Sadowska, "Wire length prediction based clustering and its application in placement", Design Automation Conference, pp.800-805, 2003.

[10] V.N.Kravets and P.Kudva, "Understanding metrics in logic synthesis for routability enhancement", ACM Intl. Workshop on System-Level Interconnect Prediction, pp.3-5, 2003.

[11] P.Kudva, A.Sullivan and W. Dougherty, "Metrics for structural logic synthesis", Proc. of ICCAD, pp. 551-556, 2002.

[12] T.Kutzschebauch and L.Stok, "Congestion aware layout driven logic synthesis", Proc. of ICCAD, pp. 216-223, 2001.

[13] Q.Liu, B.Hu and M.Marek-Sadowska, "Wire length prediction in constraint driven placement", ACM Intl. Workshop on System-Level Interconnect Prediction, pp.99-105, 2003.

[14] J.Lou, W.Chen and M.Pedram, "Concurrent logic restructuring and placement for timing closure", Proc. of ICCAD, pp. 31-35, 1999.

[15] J.Lou, S.Krishnamoorthy and H.S.Sheng, "Estimating routing congestion using probabilistic analysis", Proc. of ISPD, pp. 112-117, 2001.

[16] M.Murofushi, T.Ishioka, M.Murakata and T.Mitsuhashi, "Layout driven re-synthesis for low power consumption LSIs", Design Automation Conference, pp. 666-669, 1997.

[17] D.Pandini, L.T.Pileggi and A.J.Strojwas, "Congestion aware logic synthesis", Proc. of DATE, pp. 664-671, 2002.

[18] M.Pedram and N.Bhat, "Layout driven technology mapping", Design Automation Conference, pp.99-105, 1991.

[19] M.Pedram and B.Preas, "Accurate prediction of physical design characteristics for random logic", Proc. of Int. Conf. on Computer Design, pp.390-393, Nov. 1989.

[20] D.Stroobandt and J.Van Campenhout, "Accurate interconnection length estimations for predictions early in the design cycle", VLSI Design, vol. 10, no. 1, pp.1-20, 1999.

[21] D.Stroobandt, P.Verplaetse and J.Van Campenhout, "Generating synthetic benchmark circuits for evaluating CAD tools", IEEE Trans. on CAD, vol.19, no.9, pp.1011-1022, 2000.

[22] H.Vaishnav and M.Pedram, "Minimizing the routing cost during logic extraction", Design Automation Conference, pp.70-75, 1995.

[23] M.Wang, X.Yang and M.Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits", Proc. of ICCAD, pp. 260-264, 2000

[24] X.Yang, R.Kastner and M.Sarrafzadeh, "Congestion estimation during top-down placement", Proc. of ISPD, pp. 164-169, 2001.

[25] VLSI CAD Bookshelf:http://gigascale.org/bookshelf/.

[26] LGSynth93 benchmarks: http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth93/